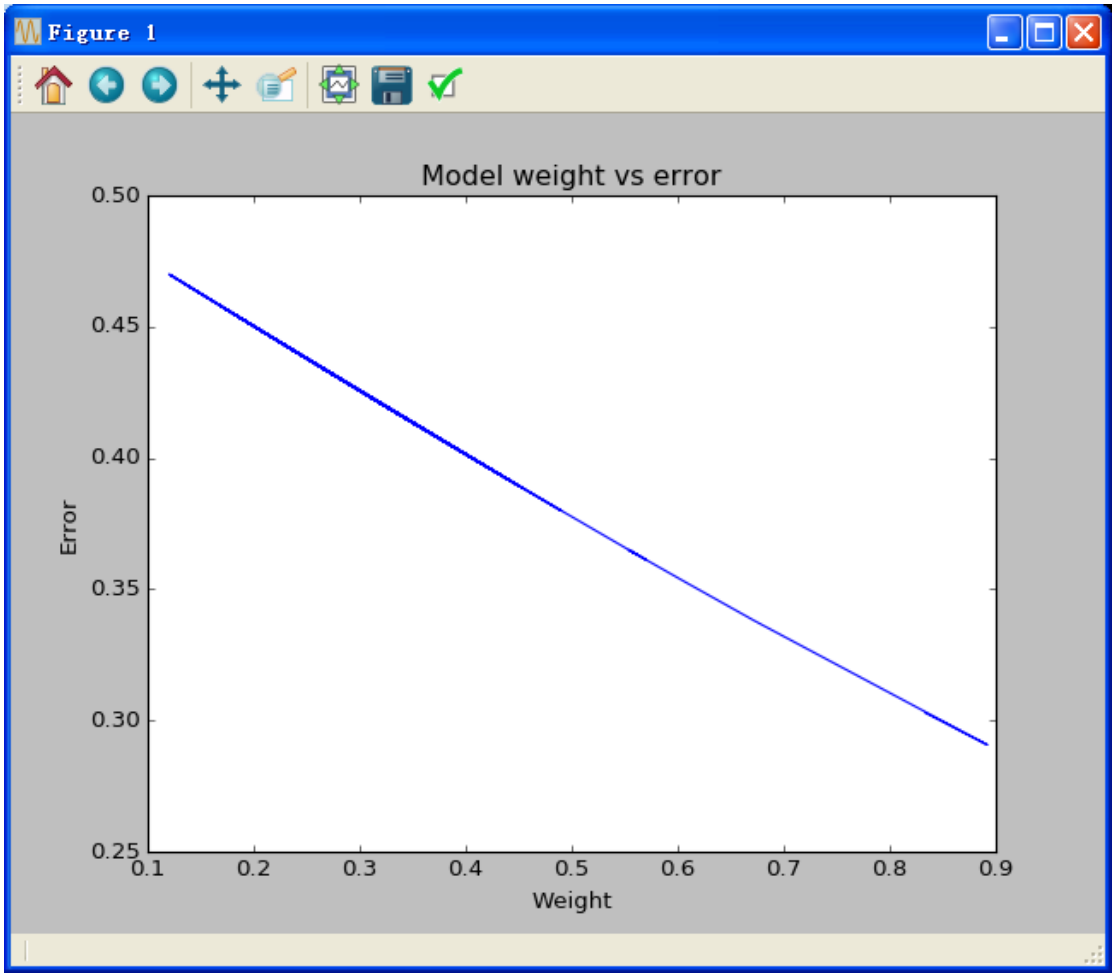
```
# Boosting 演示

from sklearn.datasets import make_classification
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import zero_one_loss
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import matplotlib.pyplot as plt
import itertools

n_f = 30                    # 特征个数
inf_f = int(0.6 * n_f)   # 60%含信息特征
red_f = int(0.1 * n_f)   # 10%冗余特征(含信息特征的线性组合)
rep_f = int(0.1 * n_f)   # 10%重复特征(随机取自含信息特征和冗余特征)

# 创建分类数据集
X,y = make_classification(
    n_samples=500,        # 实例个数
    flip_y=0.03,          # 随机抽 3%实例改变类别,产生一些噪声
    n_features=n_f,
    n_informative=inf_f,
    n_redundant=red_f,
    n_repeated=rep_f,
    random_state=7)

# 划分为训练集,验证集,测试集
X_train,X_test_all,y_train,y_test_all =
    train_test_split(X,y,test_size=0.3,random_state=9)
X_dev,X_test,y_dev,y_test =
    train_test_split(X_test_all,y_test_all,test_size=0.3,
        random_state=9)

# 单决策树模型
model = DecisionTreeClassifier()
model.fit(X_train,y_train)

# 在训练集上预测
y_pred = model.predict(X_train)
print classification_report(y_train,y_pred)
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00       181
```

```
          1      1.00      1.00      1.00      169

avg / total      1.00      1.00      1.00      350

print "Fraction of misclassification = %0.2f" %
     (zero_one_loss(y_train,y_pred)*100),"%"
Fraction of misclassification = 0.00 %

# boosting 集成
boosting = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1,min_samples_leaf=1),
    n_estimators=85,
    algorithm="SAMME",   # AdaBoosting 增强版
    random_state=9)
boosting.fit(X_train,y_train)

y_pred = boosting.predict(X_train)
print classification_report(y_train,y_pred)
             precision    recall  f1-score   support

          0      0.98      0.98      0.98      181
          1      0.98      0.98      0.98      169

avg / total      0.98      0.98      0.98      350

print "Fraction of misclassification = %0.2f" % \
     (zero_one_loss(y_train,y_pred)*100),"%"
Fraction of misclassification = 1.71 %

# 看几个模型各自随机抽取的特征,很不一样,所以各模型有变化.
for i,w in enumerate(boosting.estimator_weights_):
    print "estimator %d weight = %0.4f error = %0.4f" \
          % (i+1,w,boosting.estimator_errors_[i])

estimator 1 weight = 0.8337 error = 0.3029
estimator 2 weight = 0.8921 error = 0.2907
estimator 3 weight = 0.6730 error = 0.3378
estimator 4 weight = 0.6067 error = 0.3528
estimator 5 weight = 0.5746 error = 0.3602
....
estimator 85 weight = 0.3100 error = 0.4231

# 作图显示模型权重与错误率的关系
plt.figure(1)
```

```
plt.title("Model weight vs error")
plt.xlabel("Weight")
plt.ylabel("Error")
plt.plot(boosting.estimator_weights_,boosting.estimator_errors_)
plt.show()
```




```
# 单模型在验证集上预测
y_pred = model.predict(X_dev)
print classification_report(y_dev,y_pred)
             precision    recall  f1-score   support

          0       0.60      0.73      0.65        51
          1       0.67      0.54      0.60        54

avg / total       0.64      0.63      0.63       105

print "Fraction of misclassification = %0.2f" % \
     (zero_one_loss(y_dev,y_pred)*100),"%"
Fraction of misclassification = 37.14 %
```

```
# 集成模型在验证集上预测
y_pred = boosting.predict(X_dev)
print classification_report(y_dev,y_pred)
             precision    recall  f1-score   support

          0       0.77      0.86      0.81        51
          1       0.85      0.76      0.80        54

avg / total       0.81      0.81      0.81       105

print "Fraction of misclassification = %0.2f" % \
     (zero_one_loss(y_dev,y_pred)*100),"%"
Fraction of misclassification = 19.05 %

# 左图
no_estimators = range(20,120,10)
misclassy_rate = []
misclassy_rate_dev = []
for n in no_estimators:
    boosting = AdaBoostClassifier(
        DecisionTreeClassifier(max_depth=1,min_samples_leaf=1),
        n_estimators=n,
        algorithm="SAMME",
        random_state=9)
    boosting.fit(X_train,y_train)
    y_pred = boosting.predict(X_train)
    y_pred_dev = boosting.predict(X_dev)
    misclassy_rate.append(zero_one_loss(y_train,y_pred))
    misclassy_rate_dev.append(zero_one_loss(y_dev,y_pred_dev))

plt.figure(2)
plt.title("No estimators vs Misclassification rate")
plt.xlabel("No of estimators")
plt.ylabel("Misclassification rate")
plt.plot(no_estimators,misclassy_rate,label='Train')
plt.plot(no_estimators,misclassy_rate_dev,label='Dev')
plt.legend()
plt.show()
```

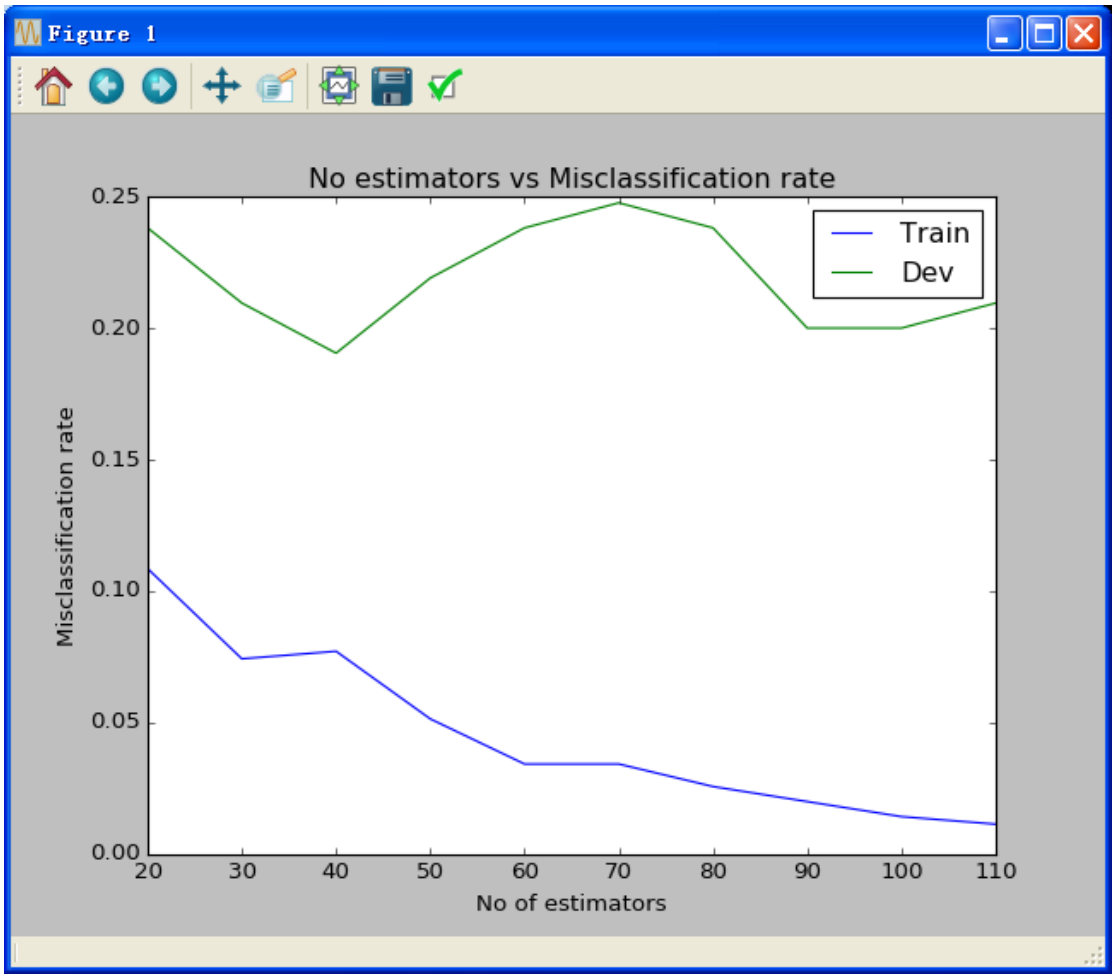


Figure 1
No estimators vs Misclassification rate
Train
Dev
Misclassification rate
No of estimators
0.25
0.20
0.15
0.10
0.05
0.00
20
30
40
50
60
70
80
90
100
110